1.2 Problem Description

UPDATE 2015-02-12: The names of some requested functions have changed. Please check that your provides match the current problem set.

Implement a game similar to the classic [JezzBall](http://en.wikipedia.org/wiki/JezzBall) game. [See here](http://onemorelevel.com/game/jezzball) to get an *approximate*idea of the gameplay mechanics.

Implement your game using 2htdp/image, 2htdp/universe, and [big-bang](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._big-bang%29%29). As usual, you must implement the standard functions required by big-bang, including a World data definition and an INITIAL-WORLD constant. Use the following run function to initiate your game (though you don’t need to provide run).

|  |
| --- |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (run w) |
| ([big-bang](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._big-bang%29%29) w |
| ([on-tick](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._on-tick%29%29) next-world) |
| ([on-key](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._on-key%29%29) key-handler) |
| ([on-mouse](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._on-mouse%29%29) mouse-handler) |
| ([on-draw](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._on-draw%29%29) render) |
| ([stop-when](http://docs.racket-lang.org/teachpack/2htdpuniverse.html#%28form._world._%28%28lib._2htdp%2Funiverse..rkt%29._stop-when%29%29) end? render-last))) |

1.2.1 Balls

The game consists of some number of balls bouncing around a 400*x*400 canvas. The balls have a constant horizontal and vertical velocity between 1*pixel/tick* and 10*pixels/tick* (inclusive), except when close to a wall.

If a tick would put any part of a ball past a canvas wall, then instead the ball moves just enough to be flush against that wall, like in previous assignments and examples. That axis of the ball’s velocity then changes direction to the opposite direction. A ball does not lose velocity when bouncing, so the magnitude of a ball’s velocity never changes, only the direction.

Balls do not interact with each other, so it’s possible for balls to overlap.

A ball has a radius of 20*pixels* and is rendered as a solid blue circle.

Implement and provide the following functions:

|  |
| --- |
| ; get-balls : World -> ListOf<Ball> |
| ; Returns all the balls in the given world. |
|  |
| ; mk-ball : Coordinate Coordinate Real Real -> Ball |
| ; Returns a Ball with center at (x,y), with the given velocities. |
| ; A positive x velocity is in the x-increasing direction and vice versa. |
| ; The y velocity is similar. |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (mk-ball x y vx vy) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |
|  |
| ; replace-balls : World ListOf<Ball> -> World |
| ; Replaces the Balls in the given World with the given Balls. |
|  |
| ; get-ball-x : Ball -> Coordinate |
| ; Returns the x position of the Ball's center. |
|  |
| ; get-ball-y : Ball -> Coordiate |
| ; Returns the y position of the Ball's center. |

1.2.2 Walls

The game player may insert new walls into the canvas by left-clicking the mouse (i.e.,"button-down").

All walls have a *major* and *minor* axis.

* For a *vertical* wall, the *y*-axis is the major axis and the *x*-axis is the minor axis.
* For a *horizontal* wall, the major and minor axes are the opposite that of a vertical wall.

A wall starts with a length of 2*pixels* along its major axis. A wall typically grows by 16*pixels/tick* along its major axis. If either end of the wall’s major axis hits the canvas edge or another perpendicular wall, it should stop growing. However, if one end hits a wall, but the other does not, then the unobstructed end should continue to grow, at 8*pixels/tick*. On a tick, if any end of a wall’s major axis would exceed a perpendicular wall or canvas edge, then that end grows by just enough to be flush against the closest wall or edge.

When both ends of a wall’s major axis hit other walls, that wall is no longer *active* (i.e., it stops growing). There may be *at most* one active wall on the canvas at any time.

If a ball hits an active wall, the game is over. Otherwise, a ball should bounce off inserted walls the same way it bounces off the canvas edge.

To simplify your computations:

* Assume that walls are lines with no thickness along its minor axis.
* You may treat canvas edges and inserted walls homogeneously.

However, *render* a wall as solid brown rectangle, with a minor axis width of 4*pixels*.

A wall may only be inserted into a section of the canvas where balls can reach. When a section of the canvas becomes unreachable, it should be colored yellow. (You may leave the walls rendered in the unreachable parts of the canvas. Don’t worry about connecting unreachable sections of the canvas into one contiguous yellow polygon, as in the sample game linked above.

When the game begins, a click inserts a vertical wall. Pressing the space bar switches to inserting horizontal walls. The game should display text to indicate which orientation wall would be inserted on the next click.

1.2.3 Scoring and Leveling

A game’s score is determined by the percentage area of the blocked off sections of the canvas, in other words, the percentage of the canvas unreachable by any ball. The areas is are pixels. To compute the score, first compute this percentage, multiply by 100, and the use [round](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._round%29%29) to get an integer score.

|  |
| --- |
| ; score : World -> Natural |
| ; Returns the current score. |
|  |
| ; level : World -> Natural |
| ; Returns the current level. |

A game begins with no walls and one ball at level 1. The ball should have a randomly generated position completely within the canvas (flush against a canvas edge is considered completely within the canvas) and constant *x* and *y* velocities between 1 and10*pixels/tick* (inclusive). Since the ball can reach all parts of the canvas, the initial score is 0.

A player then tries to insert walls to increase the score. If the score reaches or exceeds the*goal score* for the level, the game advances to the next level. Each new level should start with no walls and a number of randomly generated balls equal to the level number.

A goal score is computed as 50 plus five times the level number. The goal score should not exceed 90.

The game should display the current level, score, and goal score at all times.

As mentioned previously, the game ends when a ball collides with an active wall. The game should display GAME OVER when the game is over.

Here are some gameplay scenes.



